Bonus

Profesor Camilo Rocha

Sebastian Castro Obando
(8965484)

2024

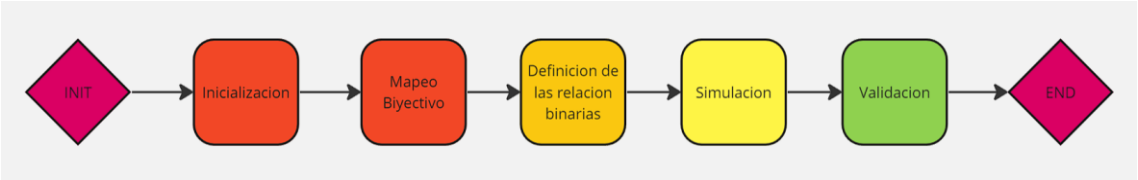
Serialización en el documento

En el documento “A formal library of set relations”[1], la serializacion se utiliza para especificar las relaciones sincrónicas mediante relaciones asíncronas, esto debido a que los sistemas de reescritura normalmente buscan implementar la máxima concurrencia de reglas de reescritura mediante reescritura asíncrona, es decir, que la serialización permite especificar relaciones sincrónicas mediante relaciones asíncronas, lo que da camino a una estructura computacional adecuada para definir y ejecutar operaciones de conjuntos y sus derivaciones de lenguajes en los lenguajes programación, que para el contexto aplicado en deGroot que hace gran uso de estructuras de datos y arboles con teoría de conjuntos, viene bastante bien.

Detalles del procedimiento de serialización

- **Sistema de representación de reglas:** En este caso, los nodos representan reglas de reescritura y las aristas representan las relaciones entre las reglas
- **Conversión de relación asincrónicas en sincrónicas:** Para lograr esto, se aplica un algoritmo de transición de un paso, en donde el objetivo es transformar una relación asíncrona en una relación sincrónica que sigue el orden de ejecución de las reglas de reescritura.
- **Sistema de relaciones sincrónicas:** Esto se puede describir como un conjunto representa todas las relaciones entre reglas de reescritura en la que una regla puede ejecutarse antes que otra.
- **El resto de las reglas ya tienen que ver en como se representan estas reglas en función de un grafo acíclico dirigido y en como se han de interpretar las trazas de forma binaria, para por último dar cavidad a la simulación y la validación del mismo.**

Diagrama



Procedimiento del algoritmo

tiene como parámetros un conjunto arbitrario T, una relación binaria → sobre U (que hace referencia a la familia de todos los conjuntos finitos sobre T), una estrategia S para →, que produce una relación R → S que simula correctamente lo cual complementa la relación sincrónica → S

En base a lo anterior se tiene lo siguiente

- El conjunto T realiza una biyeccion tipo “MARK” que mapea términos de T a T’. En donde la familia de conjuntos finitos tanto de T como de T ∪ T’, nos genera la relación U y U’. Esta biyeccion permite que los elementos de U se extiendan de forma biyectiva a los elementos de U’.
- La relación binaria → en U ⊙ se define de tal manera que A → C’ si y solo si A → B y C = mark(B) (Cabe recalcar que esto ha de ser cierto A ∈ U y C ∈ U’).

- La relación R → S en U se define de tal manera que ⟨A;B⟩ ∈ R →,s si y solo si B = (A\A') ∪ mark−1(B'), donde S(A) = {A1...An}, A' = ∪1≤ i ≤n Ai

En ultimas el teorema establece que la relación R → S, construida por el procedimiento de serialización, simula correctamente y completamente la relación sincrónica → S es decir, R → S = → S.

En base a todo lo anterior podemos evidenciar como el propósito del algoritmo es serializar sus datos mediante método asíncronos y sincrónicos, ya que esto permite generar relaciones semánticas entre agentes gracias a la teoría de conjuntos.

Posibles aplicaciones del algoritmo:

- La más lógica y posible es la aplicación en sistemas de autómatas que simulen correcta y completamente sistemas concurrentes, lo que es útil en la teoría de autómatas y en la lógica forma.
- En la ingeniera de software general, podría servir en la verificación de errores y la correlación de datos gracias a su serialización
- Puede ser de gran utilidad para convertir modelos de lenguajes formales sincrónicos en modelos asíncronos, lo que daría cavidad a una mejor manipulación de estos modelos en sistemas de reescritura asíncrona. Aunque se podría decir que la plataforma Rodin ya implementa parte de esta Teoría.
- Pero en general su aplicación va por el análisis de sistemas, incluyendo la verificación y simulación de sistemas concurrentes

Especificaciones del algoritmo (Entrada, necesidades).

- Un conjunto finito de términos T, representados por minúsculas a, b, c. Es decir, un alfabeto.
- Un conjunto U de conjuntos finitos de T, es decir, U ⊆ 2^T.
- Una relación binaria → en U, representada por una flecha entre dos elementos de U.
- Una operación binaria | en U, que mapea elementos en U × U a elementos en U. (El símbolo "|", para este caso representa la unión de conjuntos).
- Un conjunto E ⊆ U × U de elementos en U para los cuales se quiere verificar la composición de la relación "→".

Especificaciones del algoritmo (Salida).

- Se en cargade verificar si la relación binaria → en U es composicional, es decir, si para todos los pares (X, Y) ∈ E, existe un conjunto Z tal que (X → Y) = Z. Si la relación → es composicional
    - Un conjunto Z, tal que (X → Y) = Z, para todos los pares (X, Y) ∈ E.
    - Una descripción formal de las relaciones y operaciones que definen la relación composicional.
- En caso de que la relación no sea composicional, el algoritmo no genera nada, indicando al lenguaje que en efecto, no es composicional.

Posible complejidad del algoritmo

En general la complejidad temporal dependerá de la implementación específica y del tamaño de los conjuntos involucrados y como bien sabemos el algoritmo involucra operaciones como búsqueda, unión y comparación de conjuntos, las cuales tienen una complejidad temporal

polinomial en el peor de los casos. Por lo que podríamos afirmar que su complejidad es O(n*m), donde n y m denotan una relación de conjuntos.

Técnicas de programación evidenciadas

Realmente no encuentro una forma de relacionar el algoritmo de serialización con alguna de las técnicas de programación vistas en el curso ya que en su mayoría este lo que usa es la teoría de lenguajes formales, la lógica y la computación concurrente. Si analizamos los métodos por separados, tenemos que es imposible que haga uso de Greedy ya que el algoritmo hace uso de la reversibilidad, por otro lado, en algoritmo no divide el problema en subproblemas, sino que va construyendo la solución mientras se ejecuta y en ese mismo orden de ideas tampoco hace uso de programación dinámica.

Serialización en el contexto del modelo de Groot

A groso modo el articulo "Unified Opinion Dynamic Modeling"[2] nos indica que el modelo de De Groot asume que las opiniones se forman en base a la evidencia recibida y se actualizan a medida que se recibe más evidencia, por lo tanto planteemos lo siguiente.

- Participante 1: (Producto 1) - Agradezco al fabricante, porque mi hijo ama el producto.
- Participante 2: (Producto 1) - Agradezco al fabricante, pero tengo la sensación de que este producto podría mejorar un poco.
- Participante 1: (Producto 1) - Tengo unas dudas sobre la calidad del producto, pero no me gusta mucho criticar a los fabricantes.
- Participante 2: (Producto 1) - Estoy convencido de que el producto tiene ciertos defectos, pero creo que podría mejorar aún más.
    - ❖ Identificamos el conjunto T de participantes (Participante 1 y Participante 2) y el conjunto U de productos (Producto 1).
    - ❖ Se Define una estrategia s para las opiniones de los participantes, donde s(A) = {A1, A2, A3, A4}, A1 es la opinión del Participante 1 en el Producto 1 al comienzo, A2 es la opinión del Participante 1 en el Producto 1 después de que la calidad del producto empeore un poco, A3 es la opinión del Participante 1 en el Producto 1 después de que el Participante 1 tenga algunas dudas sobre la calidad del producto, y A4 es la opinión del Participante 1 en el Producto 1 después de que el Participante 1 cambie de opinión sobre el producto.
    - ❖ Si en base a lo anterior aplicamos el algoritmo de serialización con el conjunto T, la relación → sobre U, y la estrategia S. El resultado es la relación R →,S, que correctamente y completamente simula la relación → S entre las opiniones de los participantes y las posibles calidades de los productos.

Bibliografía

- [1] Rocha, C., Muñoz, C., & Dowek, G. (2011). A formal library of set relations and its application to synchronous languages. Theoretical Computer Science, 412(37), 4853-4866. https://doi.org/10.1016/j.tcs.2011.01.027
- [2] Rocha, C., Ramirez, C., Olarte, C., Valencia, F. (2024). Unified Opinion Dynamic Modeling as Concurrent Set Relations in Rewriting Logic ⋆ Javeriana Cali. https://arxiv.org/pdf/2402.09021